4. Data Frame, Factor, Formula

Data Frame (R's fundamental data structure)

A `data.frame` is (≈) a list of equal-length vectors.

e.g. `mtcars` is a built-in `data.frame`: `mtcars`, `?mtcars`, `str(mtcars)`, `summary(mtcars)`.

Factor

A *factor* represents a vector of categorical values. Categorical data must be converted to factors for R's summary, plotting, and modeling functions to work correctly.

`factor(x, levels, labels = levels)` makes a factor from vector `x`, using levels in `levels` (or the unique strings in `as.character(x)` by default), using optional `labels` to make the categories more readable. e.g.

```
m = mtcars          # work on a copy
m$vs = factor(mtcars$vs, labels=c("V", "straight"))
m$am = factor(mtcars$am, levels=c(0, 1 ,2), labels=c("auto", "man", "CVT"))
# suppose level 2 and label "CVT" are useful even though they are not in mtcars
str(m)
summary(m) # now categorical variables are handled well
```

`table(...)` makes a contingency table of counts of each combination of factors in `....` e.g. `table(m$vs)`, `table(m$vs, m$am)`

Data frame manipulation examples

```
m$mpg               # mpg column
m[ , 1]             # all rows, 1st column (mpg again)
m[1:3, 1:3]         # rows 1:3, columns 1:3
dim(m)              # dimensions
n.rows = dim(m)[1]
n.cols = length(m)  # or dim(m)[2]
tail(m)
rownames(m)[n.rows - 2] = "Monica's present"
m$hp[30] = 25
M = median(m$hp)
mean(m$mpg[m$hp > M])      # high-power mileage
mean(m$mpg[m$hp < M])      # low-power mileage
m$price = 1000*(1:n.rows)  # add column
m$vs = NULL                # delete column
sorted = m[order(m$cyl, m$disp), ] # sort by cyl, then by disp
```

(Data frame) File input and output (and ".csv" for Excel)

- `write.table(x, file = "", ...)` writes `x` to `file`. Variants include `write.csv(x, file = "")`.

  e.g. `write.csv(m, file = "mtcarsMonica.csv")` saves `m` (our corrupted `mtcars`) as comma-separated values (csv)

- `table = read.table(...)` reads from a file into a `data.frame`. Variants include

  - `table = read.csv(file, header = TRUE, row.names = 1)` for a `file` of comma-separated values with a header row of column names and a first column of row names; e.g.

    `monica = read.csv("mtcarsMonica.csv", row.names = 1)`
  - `table = read.csv(file, header = FALSE, col.names = c(...), row.names = c(...))` for a `file` of unlabeled data

Formula

A *formula* of the form `y ~ model` indicates that `y` depends on the variables in `model`. e.g. Here's a preview of the use of formulas in the coming handouts on graphics and regression.

- Here's a lousy boxplot that obscures the dependence of flower length on flower species:

```
flowers = read.csv("flowers.csv")
str(flowers) # note the factor
boxplot(flowers$Flower.Length,
        main="Flower Length Without Regard for Species", ylab="Length (mm)")
```

  Improve the graph: `boxplot(formula, ...)` makes multiple plots of data specified by `formula`. e.g. This triple plot reveals the dependence of length on species as a grouping variable:

```
boxplot(flowers$Flower.Length ~ flowers$Species,
        main="Flower Length by Species", xlab = "Species", ylab="Length (mm)")
```

- Here are similar examples using `mtcars`:

```
boxplot(m$disp)
boxplot(m$disp ~ m$am)
```

- We'll use formulas in linear regression soon:

  - `y ~ x` indicates that y depends linearly on x, as in the simple linear regression model, $y = a_1 + a_2 x$
  - `y ~ x1 + x2 + x3 + x1*x2` indicates that y depends linearly on x_1, x_2, x_3, and $x_1 \cdot x_2$, as in the multiple linear regression model, $y = a_1 + a_2 x_1 + a_3 x_2 + a_4 x_3 + a_5 x_1 \cdot x_2$.